所属类别	2022 年“华数杯”全国大学生数学建模竞赛	参赛编号
研究生		CM2205270

基于混合整数规划模型及 Gurobi 求解器的水下机器人组装计划研究

摘要

自来水管道清理机器人（CWPR）能够代替或辅助人去完成自来水管道垃圾清理任务，不仅提高自来水的品质，还能够保证水流畅通，因而越来越受到用户关注和青睐，研究 WPCR 装置的生产组装优化问题具有一定的实际应用价值。

针对问题一，本文以单周总成本最小为期望，设定各组件生产准备成本和组件储存成本累计和最小为目标函数，设置第t天生产i组件数目x_{ti}，第t天使用i组件数目u_{ti}，第t天库存i组件数目s_{ti}以及第t天是否生产i组件p_{ti}作为决策变量。建立 WPCR 需求约束、每日 A、B、C 生产总工时限制约束、第t天i组件库存数目s_{ti}约束、每周开始和结束无组件库存约束、第t天组件j的生产数目和使用约束、WPCR 的使用数目约束等约束条件，建立混合整数线性规划模型，通过 python 编程结合 Gurobi 软件求解得出在总成本最小期望下，每周 7 天的生产计划。在采取最优策略时，每周最小成本为 6259 元，其中生产准备费用共计 4800 元，库存费用共计 1459 元。

针对问题二，在连续多周生产情况下，需要考虑统筹规划。要求组件提前一天生产入库才能组装，基于问题一模型，将组件使用数目约束式中x_{ti}一项剔除。此外，将每周开始和结束时的库存约束表达式修改为每周开始各组件库存等于上周结束时各组件库存，即$s_{0i} = s_{7i}$。建立考虑统筹规划的模型，利用 Gurobi 软件求解。采取最优策略时，每周最小成本为 177207 元，其中生产准备费用共计 3600 元，库存费用共计 171922 元。问题二中模型由于组件需要提前一天生产入库，故产生了大量的库存费用。

针对问题三，在考虑停工检修的情况下，基于问题二模型增加决策变量c_t指示是否停工检修以及v_t表示检修情况下的动态变化总工时限制。修改增加总工时限制约束、停工检修次数约束、检修间隔约束、生产工时限制放宽约束等，并将原本非线性约束的生产工时限制放宽约束，转换为线性约束，建立规划模型，利用 Gurobi 软件求解。在采取最优策略时，7 次停工检修的时间分别是第 24、30、36、74、126、132、210 天，总成本最优近似解为 3762337 元，优化间隙 gap 为 0.25%。

针对问题四，考虑未来一周需求数未知的情况，基于问题二模型，加入未来 WPCR 需求预测器，建立新的模型优化未来 WPCR 需求预测器参数，根据优化的未来 WPCR 需求预测器对第 31 周的需求进行预测，然后利用问题二模型进行求解，得到未来一周的生产计划。为方便处理，模型假设每周日结束时给下周一预留的组件是恒定量，该恒定量由模型优化得到。在参数优化部分，文章采样题目给出的历史数据，得到多条“历史数据-预测数据”样本，对采样数据施加 95%以上天数 WPCR 订单正常交付，85%以上周数 WPCR 订单正常交付的约束，保证预测的效果满足题目要求。在生产计划求解阶段，文章采用预测得到的 WPCR 需求和优化的预留的组件量，得到成本最低的生产计划。除此之外，由于建立的优化预测器参数和预留的组件量的模型过于复杂，难以得到可行解，求解效率低，本文将参数优化部分模型进行简化，建立两个模型分别优化未来 WPCR 需求预测器和预留的组件量，提高求解效率。

关键词：线性规划；混合整数规划模型；有瓶颈设备多级生产计划；Gurobi 优化求解器

一、 问题重述

1.1 问题背景

自来水管道清理机器人（Water pipe cleaning robot，简称 WPCR）是一种可在水下移动、具有视觉和感知系统、通过遥控或自主操作方式、使用机械臂代替或辅助人去完成自来水管道垃圾清理任务的装置。运用这种装置能够及时清理管道，既可提高自来水的品质，也能够保证水流畅通，因而越来越受到水务公司和家庭住户的青睐。

某工厂生产的 WPCR 装置需要用 3 个容器艇（用 A 表示）、4 个机器臂（用 B 表示）、5 个动力系统（用 C 表示）组装而成。每个容器艇（A）由 6 个控制器（A1）、8 个划桨（A2）和 2 个感知器（A3）组成。每个机器臂（B）组成比较复杂，简单可划分为 2 个力臂组件（B1）和 4 个遥感器（B2）组成。每个动力系统（C）由 8 个蓄电池（C1）、2 个微型发电机（C2）和 12 个发电螺旋（C3）组成。也就是说组装一个完整的 WPCR 装置，需要 3 个容器艇（A），包括 18 个控制器（A1）、24 个划桨（A2）以及 6 个感知器（A3）。组装一台 WPCR 需要的其他部件数以此类推。组装 WPCR 所需要的产品统称为组件，包括 A 和 A1、A2、A3，B 和 B1、B2，C 和 C1、C2、C3。

该工厂每次生产计划的计划期为一周（即每次按照每周 7 天的订购数量实行订单生产），只有最终产品 WPCR 有外部需求，其他组件不对外销售。容器艇（A）、机器臂（B）、动力系统（C）生产要占用该工厂最为关键的设备，因而严格控制总生产工时。A、B、C 的工时消耗分别为 3 时/件、5 时/件和 5 时/件，即生产 1 件 A 需要占用 3 个工时，生产 1 件 B 需要占用 5 个个工时，生产 1 件 C 需要占用 5 个工时。

为了顺利生产 WPCR，工厂在某一天生产组件产品时，需要付出一个与生产数量无关的固定成本，称为生产准备费用。比如第一天生产了 A，则要支付 A 的生产准备费用，若第二天再生产 A，则需要再支付 A 的生产准备费用。如果某一天结束时某组件有库存存在，则工厂必须付出一定的库存费用（与库存数量成正比）。另外，按照工厂的信誉要求，目前接收的所有订单到期必须全部交货，轻易不能有缺货事件发生。

1.2 目标任务

本次研究的目标任务如下：

问题一：假设该工厂第一天（周一）开始时没有任何组件库存，并且要求第 7 天（周日）结束后不留下任何组件库存。每天采购的组件马上就可用于组装，组装出来的组件也可以马上用于当天组装成 WPCR。要求在总成本最小条件下，制定每周 7 天的生产计划。

问题二：实际生产中，组件 A、B、C 需要提前一天生产入库才能组装 WPCR，A1、

A2、A3、B1、B2、C1、C2、C3 也需要提前一天生产入库才能组装 A、B、C。在连续多周生产情况下，需要统筹规划。比如在周一生产 WPCR 前一天（上周周日）必须事先准备好组件库存，而且在本周日必须留下必要的组件库存用以保障下周一的生产。在给定每周的 WPCR 需求和关键设备工时限制以及每次生产准备费用和单件库存费用数据情况下，要求制定每周 7 天的生产计划以使总成本最低。

问题三：在问题二的情况下，保障生产的持续性，工厂需要在 30 周 210 天里必须设置 7 次停工检修，每次检修时间为 1 天。检修之后关键设备生产能力有所提高，检修后的第一天 A、B、C 生产总工时限制将会放宽 10%，随后逐日减少放宽 2%的比例，直至为 0。检修日的订单只能提前安排生产，当天不能生产任何组件。假设每周的关键设备工时限制以及每次生产准备费用和单件库存费用数据不变，任意两次检修之间要相隔 6 天以上。在给定 30 周的 WPCR 外部需求数据情况下，求检修日放在哪几天使得总成本最小。

问题四：在生产实际中，在未知 WPCR 外部需求订单的前提下，公司需要有一个稳妥的单周生产计划。在问题二的情况下，给定历史周订单数据，在不知未来某周 7 天订单数且继续追求周总成本最小的前提下，如何制定周生产计划，既能够保障每天的 WPCR 订单均以 95%以上的概率保证正常交付，又能够以 85%以上的概率保证整周的 WPCR 订单能正常交付。

二、 问题分析

WPCR 装置由 3 个容器艇（用 A 表示）、4 个机器臂（用 B 表示）、5 个动力系统（用 C 表示）组装。WPCR 组装成本由生产准备成本和组件储存成本构成，整理各组件工时消耗、生产准备费用、单位库存费用等参数如图 1 所示。

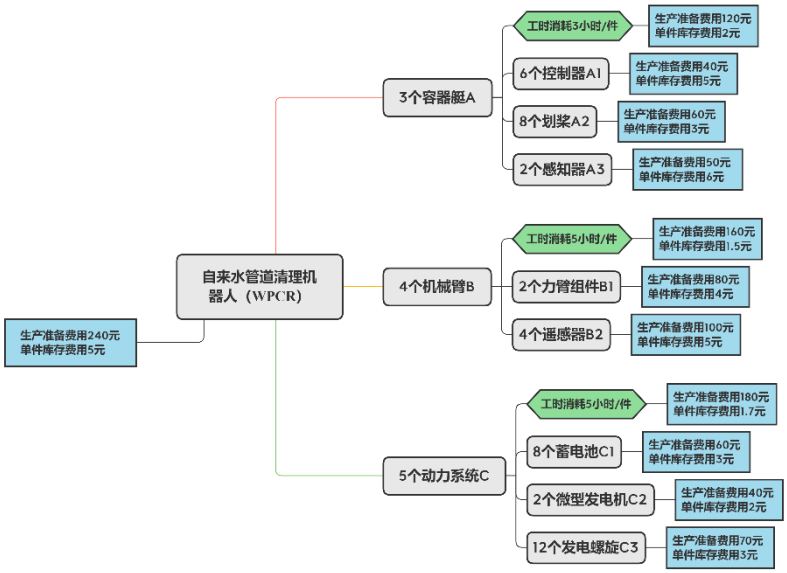

图 1：各组件参数整理

2.1 问题一的分析

问题一属于混合整数线性规划问题[1]，按照表 6 中 WPCR 到 C3 顺序，共有 WPCR、A、A1、A2、A3、B、B1、B2、C、C1、C2、C3，总计十二类产品组件。针对问题一，我们以第t天生产各组件的数目、第t天使用各组件的数目、第t天各组件的库存数目以及第t天是否生产某组件的指示变量作为决策变量，选取生产准备成本和组件储存成本构成的 WPCR 一周组装成本作为目标函数，建立数学模型求解目标函数的最小值[2]。利用 Gurobi 求解最优总成本期望下的单周生产计划。

2.2 问题二的分析

问题二属于问题一的变形，要求组件A、B、C需要提前一天生产入库才能组装WPCR，A1、A2、A3、B1、B2、C1、C2、C3 也需要提前一天生产入库才能组装 A、B、C。因此需要将问题一求解模型中的组件生产所需其他组件的数目约束条件修改为$t-1$日的j组件库存数$s_{t-1,j}$大于等于t日j组件使用数目u_{tj}。同时，模型的边界条件也发生了变化，在问题一中，周一开始和周末结束工厂均没有组件库存。在问题二中需要在本周日留下必要的组件库存用以保障下周一的生产，因此每周开始和结束时的库存约束表达式也应进行修改。利用 Gurobi 软件求解最优总成本期望下的单周生产计划。

2.3 问题三的分析

问题三要求在问题二的基础上，在 210 天里设置单次检修时间为 1 天的 7 次停工检修。增加决策变量c_t指示工厂是否在第t天停工检修，以及决策变量v_t表示考虑工厂检修提高关键设备生产能力情况下，第t天生产组件的总工时限制[3]。在问题二的约束条件中，将原本常量的总工时限制修改为动态变化的约束变量，将是否停工检修的约束添加进生产相关约束条件，同时需要对总停工检修次数和任意两次停工检修时间间隔进行约束，最后对生产工时限制放宽约束。利用 Gurobi 求解最优总成本期望下的检修日时间安排。

2.4 问题四的分析

问题四要求在问题二的基础上，根据历史周订单数据，计算出未来某周 7 天订单数的预测需求。考虑未来一周需求数未知的情况，基于问题二模型，加入未来 WPCR 需求预测器，建立新的模型优化未来 WPCR 需求预测器参数，根据优化的未来 WPCR 需求预测器对第 31 周的需求进行预测，然后利用问题二模型进行求解得到未来一周的生产计划[4]。

三、 模型假设

1、假设生产组装过程中组件无损坏、误装等人工消耗。
2、不考虑组件存放空间需求，假设工厂可容纳组件库存无上限。
3、不考虑生产的直接成本（如原材料成本、人力成本、电力成本等）。
4、假设除生产停工外其它时间，工厂设备均能正常运转。

四、 主要符号说明

符号	符号说明	单位
t	时间，$1 \le t \le 7$表示周一到周日	天
i	组件类型	
j	组件类型，与i含义相同，用于区分不同组件	
x_{ti}	第t天生产i组件的数目	件/天
u_{ti}	第t天使用i组件的数目	件/天
s_{ti}	第t天库存i组件的数目	件/天
p_{ti}	第t天是否生产i组件的指示变量	
R_t	第t天的 WPCR 的数目需求	件/天
TU_t	第t天生产组件的总工时限制	工时/天
TC_i	生产组件i的工时耗费	工时/件
CP_i	生产组件i的生产准备费用	元/天
CS_i	存储组件i的单件库存费用	元/件/天
RP_{ij}	生产组件i对组件j的数目需求	
c_t	决策变量，用于指示工厂是否在第t天停工检修	
v_t	检修时间段第t天生产组件的总工时限制	工时/天
S	停工天数	天
DS	两次停工最小间隔	天
DTU_t	一次停工对t天后的工时影响为放宽时间限制	
DT	一次停工影响范围	天
d_t^l	第l样本的完成指示变量	
w^l	第l样本的整周完成指标变量	
z_t^l	第l样本第t天的预测需求	件/天
$\bar{s}_t^l$	第l样本第t天在预测需求下生产的 WPCR 实际分配库存	件/天

注：此为本文的主要符号说明，其它符号解释详见正文部分。

五、 模型建立与求解

5.1 问题一的模型建立与求解

5.1.1 模型的准备和建立

问题一要求我们在总成本最小的期望下，制定每周 7 天的生产计划。题目要求周一开始和周末结束时，均没有任何组件库存，且当天采购、组装的组件马上可投入组装、使用。本问题属于混合整数规划问题，选取生产准备成本和组件储存成本构成的 WPCR 一周组装成本最优作为目标函数，下面开始混合整数线性规划模型的建立[5]。

设下标t，取值范围为：$1 \le t \le 7$，表示时间周一到周日。设下标i，取值范围为：$1 \le i \le 12$，表示题目中所给的 WPCR、A、A1、A2、A3、B、B1、B2、C、C1、C2、C3 总计十二类产品组件类型。设下标j，取值范围为：$1 \le j \le 12$，j与i含义相同，用于区分不同组件。

下面开始设决策变量，设第t天生产i组件的数目为x_{ti}，$x_{ti} \in N$，N为自然数集；第t天使用i组件的数目为u_{ti}，$u_{ti} \in N$；第t天库存i组件的数目为s_{ti}，$s_{ti} \in N$；第t天是否生产i组件的指示变量记为p_{ti}，$p_{ti} \in \{0,1\}$。

生产组件i对组件j的数目需求记为RP_{ij}；生产组件i的工时耗费为TC_i。需要注意的是，题目中$TC_2 = 3$，$TC_6 = 5$，$TC_9 = 5$，分别表示组件 A，B，C 的工时耗费，其余组件类型的$TC_i = 0$；题目中$RP_{12} = 3$，$RP_{16} = 4$，$RP_{19} = 5$，$RP_{23} = 6$，$RP_{24} = 8$，$RP_{25} = 2$，$RP_{67} = 2$，$RP_{68} = 4$，$RP_{9,10} = 8$，$RP_{9,11} = 2$，$RP_{9,12} = 12$，其余为 0。

（1） 目标函数

题目要求一周总成本最小，故目标函数设定为各组件生产准备成本和组件储存成本累计和最小：

$$\min \sum_t \sum_i CP_i p_{ti} + CS_i s_{ti} \tag{1}$$

其中，CP_i表示生产组件i的生产准备费用，p_{ti}表示第t天是否生产i组件的指示变量。CS_i表示存储组件i的单件库存费用，s_{ti}表示第t天库存i组件的数目。

（2） 约束条件

题目给定 WPCR 每天的需求，设第t天的 WPCR 的数目需求为R_t，则有 WPCR 需求约束：

$$x_{t1} + s_{t-1,1} \ge R_t \tag{2}$$

其中，x_{t1}表示第t天组装的 WPCR 数目，$s_{t-1,1}$表示第$t-1$天库存 WPCR 的数目，R_t

表示第t天的 WPCR 的数目需求。

题目给定每日 A、B、C 生产总工时限制，设第t天生产组件的总工时限制为TU_t，则有生产工时约束：

$$\sum_i TC_i x_{ti} \le TU_t \tag{3}$$

其中，TC_i表示生产组件i的工时耗费，x_{t1}表示第t天组装的 WPCR 数目。

指示变量$p_{ti} \in \{0,1\}$，则有离散 0-1 变量和实数变量的转换约束：

$$x_{ti} \le p_{ti} M \tag{4}$$

其中，M为极大常量。

第t天库存i组件的数目s_{ti}存在库存约束关系：

$$s_{ti} = s_{t-1,i} + x_{ti} - u_{ti} \tag{5}$$

每周开始无库存，则有：

$$s_{0i} = 0 \tag{6}$$

每周结束无库存，则有：

$$s_{7i} = 0 \tag{7}$$

第t天组件j的生产数目约束：

$$x_{tj} + s_{t-1,j} \ge u_{tj}, 2 \le j \le 12 \tag{8}$$

第t天组件j的使用数目约束：

$$u_{tj} = \sum_i RP_{ij} x_{ti}, 2 \le j \le 12 \tag{9}$$

其中，RP_{ij}表示生产组件i对组件j的数目需求。

WPCR 的使用数目约束：

$$u_{t1} = R_t \tag{10}$$

（3） **建立模型**

结合目标函数(1)式以及所列约束条件(2)~(10)式，可建立混合整数线性规划模型：

$$\min \sum_t \sum_i CP_i p_{ti} + CS_i s_{ti}$$

$$s.t.\begin{cases} x_{t1}+s_{t-1,1}\geq R_t \\ \sum_i TC_i x_{ti}\leq TU_t \\ x_{ti}\leq p_{ti}M \\ s_{ti}=s_{t-1,i}+x_{ti}-u_{ti} \\ s_{0i}=0 \\ s_{7i}=0 \\ x_{tj}+s_{t-1,j}\geq u_{tj}, 2\leq j\leq 12 \\ u_{tj}=\sum_i RP_{ij}x_{ti}, 2\leq j\leq 12 \\ u_{t1}=R_t \\ x_{ti},u_{ti},s_{ti}\in N \\ p_{ti}\in\{0,1\} \end{cases} \tag{11}$$

在式(11)中，如无特殊说明，$1\leq t\leq 7$，$1\leq i\leq 12$，$1\leq j\leq 12$。

5.1.2 模型的求解

利用 python 语言编程调用 Gurobi 软件，编写程序求解本问题所建立上述模型，得到每周 7 天的生产计划以及对应的生产准备费用和库存费用。整理求解结果如表 1 所示。

表 1：问题 1 的结果

日期	WPCR 组装数量	A 组装数量	B 组装数量	C 组装数量	生产准备费用	库存费用
周一	83	249	332	416	1200	0
周二	0	0	344	0	340	221
周三	81	243	0	404	860	557
周四	0	0	0	0	0	285
周五	48	144	173	240	1200	85
周六	51	153	203	255	1200	111
周日	0	0	0	0	0	200
总和	263	789	1052	1315	6259	

5.1.3 模型结果分析

分析结果可知，在采取最优策略时，每周最小成本为 6259 元，其中生产准备费用共计 4800 元，库存费用共计 1459 元。

为验证本问题所建立模型求解最优解得到的单周生产计划的准确性和有效性。基于问题一求解结果，通过 python 编程计算在制定的单周生产计划下，A、B、C 生产总工时是否满足限制、每日组装 WPCR 的所需的 A、B、C 工件数目是否满足需求、以及每日使用的 WPCR 数是否满足需求。

编程计算在该生产计划下，单周每日 A、B、C 组件生产总工时消耗，以及每日交付 WPCR 需求后 WPCR 每日盈余，计算结果如表 2 所示。计算单周每日 A、B、C 组件交付组装 WPCR 需求后盈余，计算结果如表 3 所示。

表 2：生产工时及 WPCR 需求验证结果

日期	限制生产工时	实际生产工时	WPCR 需求数	WPCR 组装数	交付需求后 WPCR 盈余
周一	4500	4487	39	83	44
周二	2500	1720	36	0	8
周三	2750	2749	38	81	51
周四	2100	0	40	0	11
周五	2500	2497	37	48	22
周六	2750	2749	33	51	40
周日	1500	0	40	0	0

由表 2 可知，单周每日 A、B、C 组件生产总工时消耗小于单周每日 A、B、C 生产总工时限制。而每日交付需求后 WPCR 盈余均为自然数，且在周日交付需求后 WPCR 盈余为 0 件，满足题目要求的工厂周一开始时和周末结束时均没有 WPCR 组件库存。

表 3：生产 WPCR 所需 ABC 组件验证结果

日期	WPCR 组装数	交付需求后 A 盈余	交付需求后 B 盈余	交付需求后 C 盈余
周一	83	0	0	1
周二	0	0	344	1
周三	81	0	20	0
周四	0	0	20	0
周五	48	0	1	0
周六	51	0	0	0
周日	0	0	0	0

由表 3 可知，单周每日交付组装 WPCR 需求的 A、B、C 组件后，每日结束时 A、B、C 组件盈余均为自然数，且在周日交付需求后 A、B、C 组件盈余均为 0 件，满足题目要求的工厂周一开始时和周末结束时均没有任何组件库存。

5.2 问题二的模型建立与求解

5.2.1 模型的准备和建立

（1） 目标函数

本问题与问题一模型基本一致，需要在考虑统筹规划前提下，要求一周总成本最小，仍选取生产准备成本和组件储存成本构成的 WPCR 一周组装成本最优作为目标函数。目标函数设定为各组件生产准备成本和组件储存成本累计和最小：

$$\min \sum_{t} \sum_{i} CP_i p_{ti} + CS_i s_{ti}$$

（2） 约束条件

本问题要求组件 A、B、C 需要提前一天生产入库才能组装 WPCR，A1、A2、A3、B1、B2、C1、C2、C3 也需要提前一天生产入库才能组装 A、B、C。考虑在连续多周生产情况下，需要统筹规划。需要将问题一求解模型中的组件生产所需其他组件的数目约束条件修改为$t-1$日的j组件库存数$s_{t-1,j}$大于等于t日j组件使用数目u_{tj}，即第t天组件j的生产数目约束：

$$s_{t-1,j} \geq u_{tj}, 2 \leq j \leq 12 \tag{12}$$

同时，模型的边界条件也发生了变化，在问题一中，周一开始和周末结束工厂均没有组件库存。在本问题中需要在本周日留下必要的组件库存用以保障下周一的生产，因此每周开始和结束时的库存约束表达式修改为每周开始各组件库存等于上周结束时各组件库存，即：

$$s_{0i} = s_{7i} \tag{13}$$

（3） 建立模型

在问题一所建立模型基础上，结合所列修改约束条件(12)、(13)式，建立本问题考虑统筹规划时的混合整数线性规划模型：

$$\min \sum_t \sum_i CP_i p_{ti} + CS_i s_{ti}$$

$$s.t. \begin{cases} x_{t1} + s_{t-1,1} \geq R_t \\ \sum_i TC_i x_{ti} \leq TU_t \\ x_{ti} \leq p_{ti} M \\ s_{ti} = s_{t-1,i} + x_{ti} - u_{ti} \\ s_{0i} = s_{7i} \\ s_{t-1,j} \geq u_{tj}, 2 \leq j \leq 12 \\ u_{tj} = \sum_i RP_{ij} x_{ti}, 2 \leq j \leq 12 \\ u_{t1} = R_t \\ x_{ti}, u_{ti}, s_{ti} \in N \\ p_{ti} \in \{0,1\} \end{cases} \tag{14}$$

在式(14)中，如无特殊说明，$1 \leq t \leq 7$，$1 \leq i \leq 12$，$1 \leq j \leq 12$。

5.2.2 模型的求解

与问题一类似，利用 python 语言编程调用 Gurobi 软件，编写程序求解本问题所建立上述模型，得到每周 7 天的生产计划以及对应的生产准备费用和库存费用。整理求解结果如表 4 所示。

表 4：问题 2 的结果

日期	WPCR 组装数量	A 组装数量	B 组装数量	C 组装数量	生产准备费用	库存费用
周一	0	246	341	411	630	52318
周二	82	0	0	500	750	33737
周三	0	300	370	0	280	31296
周四	100	0	0	0	420	2101
周五	0	0	341	0	480	9899
周六	0	243	0	404	300	42571
周日	81	0	0	0	740	1685
总和	263	789	1052	1315	177207	

5.2.3 模型结果分析

分析结果可知，在采取最优策略时，每周最小成本为 177207 元，其中生产准备费用共计 3600 元，库存费用共计 171922 元。相比与问题一模型求解的最优策略，即每周最小成本为 6259 元（其中生产准备费用共计 4800 元，库存费用共计 1459 元），不难看出，问题一、二求解的生产准备费用比较接近，而问题二中模型由于考虑统筹规划且组件需要提前一天生产入库才能组装更高一级组件，因此产生了大量的库存费用。相比之下，问题二求解结果中的单周累计库存费用大约是问题一求解结果中的单周累计库存费用的 118 倍。

本问题所建立模型，每周开始各组件库存等于上周结束时各组件库存，即$s_{0i} = s_{7i}$。计算求出模型最优解时，查看s_{0i}，可知 WPCR、A、A1、A2、A3、B、B1、B2、C、C1、C2、C3 十二类产品组件在周末结束后库存分别为：41、0、1476、1968、492、17、682、1364、0、3288、822、4932 件。可以发现每周结束后组件 A 和组件 C 库存均为 0 件，而由于组件 A、B、C 需要提前一天生产入库才能组装 WPCR，因此周一的 WPCR 组装数量为 0 件，这个情况也正好符合求出的模型最优解。而每周结束的 WPCR 库存为 41 件，在下周一无法组装 WPCR 的情况下，恰好能满足周一 39 件 WPCR 的需求。

编程计算在该生产计划下，单周每日 A、B、C 组件生产总工时消耗，以及每日交付 WPCR 需求后 WPCR 每日盈余，计算结果如表 5 所示。计算单周每日 A、B、C 组件交付组装 WPCR 需求后盈余，计算结果如表 6 所示。

表 5：生产工时及 WPCR 需求验证结果

日期	限制生产工时	实际生产工时	WPCR 需求数	WPCR 组装数	交付需求后 WPCR 盈余
周一	4500	4498	39	0	2
周二	2500	2500	36	82	48
周三	2750	2750	38	0	10
周四	2100	0	40	100	70
周五	2500	1750	37	0	33

周六	2750	2749	33	0	0
周日	1500	0	40	81	41

由表 5 可知，单周每日 A、B、C 组件生产总工时消耗小于等于单周每日 A、B、C 生产总工时限制。而每日交付需求后 WPCR 盈余均为自然数，且在周日交付需求后 WPCR 盈余为 41 件，满足周一 39 件 WPCR 的需求，满足题目要求，一定程度验证了模型求得解的正确性。

周一组装 A、B、C 组件的数量分别为：246、341、411 件，因此需要消耗 A1、A2、A3、B1、B2、C1、C2、C3 组件数量分别为：1476、1968、492、682、1364、3288、822、4932 件。而在周末结束后 A1、A2、A3、B1、B2、C1、C2、C3 组件库存分别为：1476、1968、492、682、1364、3288、822、4932 件，恰好可以满足周一组装 A、B、C 组件需求。

表 6：生产 WPCR 所需 ABC 组件验证结果

日期	A 需求数目	B 需求数目	C 需求数目	A 组装数目	B 组装数目	C 组装数目	交付需求后 A 盈余	交付需求后 B 盈余	交付需求后 C 盈余
周一	0	0	0	246	341	411	246	358	411
周二	246	328	410	0	0	500	0	30	501
周三	0	0	0	300	370	0	300	400	501
周四	300	400	500	0	0	0	0	0	1
周五	0	0	0	0	341	0	0	341	1
周六	0	0	0	243	0	404	243	341	405
周日	243	324	405	0	0	0	0	17	0

由表 6 可知，单周每日交付组装 WPCR 需求的 A、B、C 组件后，每日结束时 A、B、C 组件盈余均为自然数，且在周日交付需求后 A、B、C 组件盈余分别为 0、17、0 件，满足周一 WPCR 的需求，满足题目要求，验证了模型求得解的正确性。

5.3 问题三的模型建立与求解

5.3.1 模型的准备和建立

本问题中，需要确定工厂在 30 周 210 天安排的 7 次停工检修时间。基于问题二所建立模型，对时间长度进行扩展，下标t范围变为$1 \leq t \leq 210$。第t天生产组件的原始总工时限制TU按周次重复扩展到 210 天，即：第二周第一天的原始总工时限制与上周第一天原始总工时限制相等，依次类推。

考虑到 7 次停工检修，每次检修时间为 1 天，检修日的订单只能提前安排生产，当天不能生产任何组件，故增加决策变量c_t，用于指示工厂是否在第t天停工检修，$c_t = 1$，表示工厂在第t天停工检修，否则正常工作。考虑到检修之后关键设备生产能力有所提

高，检修后的第一天 A、B、C 生产总工时限制将会放宽 10%，随后逐日减少放宽 2%的比例，直至为 0。增加决策变量v_t，v_t表示考虑工厂检修提高，关键设备生产能力情况下，第t天生产组件的总工时限制[6]。

（1） 目标函数

问题要求 30 周总成本最小，选取生产准备成本和组件储存成本构成的 WPCR 一周组装成本最优为目标。目标函数设定为各组件生产准备成本和组件储存成本累计和最小：

$$\min \sum_t \sum_i CP_i p_{ti} + CS_i s_{ti}$$

（2） 约束条件

本问题所建立模型，考虑工厂检修后一段时间内关键设备生产能力有所提高，故问题一、二建立模型中每日 A、B、C 生产总工时限制TU_t修改为动态变化的约束变量，即：

$$\sum_i TC_i x_{ti} \le v_t \tag{15}$$

决策变量c_t，用于指示工厂是否在第t天停工检修。决策变量p_{ti}，用于指示工厂第t天是否生产i组件。工厂第t天是否生产i组件受限于是否在第t天停工检修，即：

$$p_{ti} \le 1 - c_t \tag{16}$$

工厂在 30 周 210 天里共设置 7 次停工检修，$c_t = 1$，表示工厂在第t天停工检修，因此在时间长度$1 \le t \le 210$范围，存在约束：

$$\sum_t c_t = S \tag{17}$$

题目要求任意两次检修之间要相隔 6 天以上，两次检修间隔存在约束，即：

$$\sum_{r=t}^{t+DS-1} c_r \le 1, 1 \le t \le 211 - DS \tag{18}$$

$$\sum_{r=1}^{t-211+DS} c_r + \sum_{r=t}^{210} c_r \le 1, 212 - DS \le t \le 210 \tag{19}$$

其中，DS表示两次停工最小间隔（包括DS），题目中$DS = 6$。

v_t表示考虑工厂检修提高关键设备生产能力情况下，第t天生产组件的总工时限制。本问题模型中，存在生产工时限制放宽约束：

$$v_1 = \max\{1 + DTU_{211-r} c_r | 211 - DT \le r \le 210\} TU_1 \tag{20}$$

$$v_t = \max\left\{\begin{matrix}\max\{1 + DTU_{t-r} c_r | 1 \le r \le t-1\}, \\ \max\{1 + DTU_{t+210-r} c_r | 210 + t - DT \le r \le 210\}\end{matrix}\right\} TU_t c_r, 2 \le t \le DT \tag{21}$$

$$v_t = \max\{1 + DTU_{t-r} c_r | t - DT \le r \le t-1\} TU_t c_r, DT + 1 \le t \le 210 \tag{22}$$

其中，DTU_t为放宽时间限制，表示工厂一次停工检修对t天后的关键设备生产能力影响，即 A、B、C 生产总工时限制的放宽，影响范围为DT天，本题中$DTU_t = 0.1 -$

$0.02(t-1), 1 \le t \le DT$，$DT = 5$。

考虑到停工检修对关键设备生产能力提高，即生产总工时限制的放宽具有影响范围，根据放宽时间限制DTU_t，在本题中 5 天内不可能存在 2 次检修，即当$DT < DS$时，所以非线性约束式（20）、（21）、（22）可以转化为线性约束：

$$v_1 = \left(1 + \sum_{r=211-DT}^{210} DTU_{211-r}\, c_r\right) TU_1 \tag{23}$$

$$v_t = \left(1 + \sum_{r=1}^{t-1} DTU_{t-r} c_r + \sum_{r=210+t-DT}^{210} DTU_{t+210-r} c_r\right) TU_t, 2 \le t \le DT \tag{24}$$

$$v_t = \left(1 + \sum_{r=t-DT}^{t-1} DTU_{t-r}\, c_r\right) TU_t, DT+1 \le t \le 210 \tag{25}$$

将非线性约束式（20）、（21）、（22）可以转化为线性约束式（23）、（24）、（25），从而将本问题从非线性规划模型转为线性规划模型。

（3） 建立模型

基于问题二所建立模型，增加决策变量c_t指示工厂是否在第t天停工检修以及决策变量v_t表示考虑工厂检修时第t天生产组件的总工时限制。结合所列约束条件(15)~(25)式，建立本问题考虑工厂检修时的混合整数线性规划模型：

$$\min \sum_t \sum_i CP_i p_{ti} + CS_i s_{ti}$$

约束条件如式(26)所示，由于式(26)过长考虑排版问题，将其分为式(26-1)和式(26-2)

$$s.t. \begin{cases} x_{t1} + s_{t-1,1} \ge R_t \\ \sum_i TC_i x_{ti} \le v_t \\ x_{ti} \le p_{ti} M \\ s_{ti} = s_{t-1,i} + x_{ti} - u_{ti} \\ s_{0i} = s_{7i} \\ s_{t-1,j} \ge u_{tj}, 2 \le j \le 12 \\ u_{tj} = \sum_i RP_{ij} x_{ti}, 2 \le j \le 12 \\ u_{t1} = R_t \\ p_{ti} \le 1 - c_t \\ \sum_t c_t = S \\ \sum_{r=t}^{t+DS-1} c_r \le 1，\ 1 \le t \le 211 - DS \\ \sum_{r=1}^{t-211+DS} c_r + \sum_{r=t}^{210} c_r \le 1，\ 212 - DS \le t \le 210 \end{cases} \tag{26-1}$$

$$s.t.\begin{cases} v_1 = \left(1 + \sum_{r=211-DT}^{210} DTU_{211-r}\, c_r\right) TU_1 \\ v_t = \left(1 + \sum_{r=1}^{t-1} DTU_{t-r} c_r + \sum_{r=210+t-DT}^{210} DTU_{t+210-r} c_r\right) TU_t, 2 \le t \le DT \\ v_t = \left(1 + \sum_{r=t-DT}^{t-1} DTU_{t-r}\, c_r\right) TU_t, DT+1 \le t \le 210 \\ x_{ti}, u_{ti}, s_{ti} \in N \\ c_t, p_{ti} \in \{0,1\} \end{cases} \quad (26-2)$$

在式(26)，即式(26-1)和式(26-2)中，如无特殊说明，$1 \le t \le 210$，$1 \le i \le 12$，$1 \le j \le 12$。

5.3.2 模型的求解

本问题在问题二的基础上，增加停工检修相关决策变量，通过建模将非线性规划问题转为线性规划问题，故与问题二类似。利用 python 语言编程调用 Gurobi 软件，编写程序求解本问题所建立上述模型，得到最优总成本期望下，工厂在 30 周 210 天里设置的 7 次停工检修时间安排结果，整理求解结果如表 7 所示[7]。

表 7：问题 3 的结果（检修日期及总成本）

第 1 次	第 2 次	第 3 次	第 4 次	第 5 次	第 6 次	第 7 次	总成本
24	30	36	74	126	132	210	3762337

根据表 7 可知，在采取最优策略时，7 次停工检修的时间分别是第 24、30、36、74、126、132、210 天，总成本最优近似解为 3762337 元，优化间隙 gap 为 0.25%（Gurobi 中优化间隙 gap 定义为最优目标上下界差占最优上界的比例）。其中生产准备费用共计 47730 元，库存费用共计 3714607 元。具体的 30 周生产计划及对应的对应的生产准备费用和库存费用详见支撑材料“question3.csv”文件。

5.4 问题四的模型建立与求解

5.4.1 模型的准备和建立

本问题中，需要通过 30 周 210 天的历史周订单数据，在未知 WPCR 外部需求订单的前提下，制定一个稳妥的单周生产计划。也就是说，需要根据历史周订单数据，计算出未来某周 7 天订单数的预测需求。基于问题二模型，在最优总成本期望下，要求既能够保障每天的 WPCR 订单均以 95%以上的概率保证正常交付，又能够以 85%以上的概率保证整周的 WPCR 订单能正常交付。

这里假设每周预留给下周的组件或 WPCR 数目一定。未来一周的预测函数为$f: Y \to z$，表示从长度为m的历史需求数据到未来n天 WPCR 需求的映射，题目中$n = 7$。为计

算出未来某周 7 天订单数的预测需求，需要在历史周订单数据中采样，采样方法为：z的实际值取Z整周 7 天 WPCR 需求，Y取采样周前m天的历史数据，m为n的整数倍。按照此方法共可采样获得$30-m/7$个样本。其中，时间指示下标$1 \le t \le 7$；采样的样本的指示上标$1 \le l \le 30-m/7$。

设置决策变量d_t^l表示第l样本的完成指示变量，$d_t^l = 1$表示在第l样本中，第t天的生产满足需求，即 WPCR 存量和当天生产量可供给当天 WPCR 需求；否则当$d_t^l = 0$，当天不满足需求。决策变量w^l表示第l样本的整周完成指标变量，$w^l = 1$表示第l样本整周完成 WPCR 需求，即整周内任意一天都满足 WPCR 需求；否则，第l样本的当周不满足需求。z_t^l表示第l样本第t天的预测需求。$\bar{s}_t^l$表示第l样本第t天在预测需求下生产的 WPCR 实际分配库存。

（1） 目标函数

问题要求不知未来某周 7 天订单数且继续追求周总成本最小的前提下，制定周生产计划。也就是说要在预测的未来某周 7 天订单数需求下，追求总成本最小。目标函数设定为预测某周订单需求情况下，各组件生产准备成本和组件储存成本累计和最小：

$$\min \sum_l \sum_t \sum_i CP_i p_{ti}^l + CS_i s_{ti}^l \tag{27}$$

（2） 约束条件

本问题模型求解数据为预测数据，因此参数增加了l上标。

未来某周第l样本第t天预测 WPCR 需求约束：

$$x_{t1}^l + s_{t-1,1}^l \ge z_t^l \tag{28}$$

这里假设每周预留给下周的组件或 WPCR 数目一定，即存在约束：

$$s_{0i}^l = s_{7i}^k, 1 \le l,k \le 30-n/7 \tag{29}$$

未来某周第l样本第t天预测 WPCR 需求z_t^l，满足预测函数约束：

$$z_t^l = f(Y^l)_t \tag{30}$$

WPCR 使用量等于需求量约束：

$$u_{t1}^l = z_t^l \tag{31}$$

第l样本第t天在预测需求下生产的 WPCR 实际分配库存$\bar{s}_t^l$，在预测需求下生产的 WPCR 实际分配库存更新约束：

$$\bar{s}_t^l = \left(\bar{s}_{t-1}^l + x_{ti}^l - Z_t^l\right) d_t^l \tag{32}$$

本问题假设每周预留给下周的组件或 WPCR 数目一定，故存在初始库存一致约束：

$$\bar{s}_0^l = s_{01}^l \tag{33}$$

生产量大于库存量约束：

$$\sum_t x_{t1}^l \ge s_{01}^l \tag{34}$$

去掉该约束会产生预测需求量为 0 的不合理解。

WPCR 使用量大于实际需求量时，标记为当天满足需求的约束：

$$\bar{s}_{t-1}^{l} + x_{ti}^{l} - Z_t^l + \left(1 - d_t^l\right)M \geq 0 \tag{35}$$

95%以上的天数满足需求约束：

$$\sum_{l}\sum_{t} d_t^l \geq 0.95(210 - m) \tag{36}$$

其中公式右侧展开为：$0.95(210 - m) = 0.95 \times 7(30 - m/7)$；

当周所以天数均满足需求时，整周满足需求约束：

$$\sum_{t} d_t^l \geq 7w^l \tag{37}$$

85%以上的周次满足约束：

$$\sum_{l} w^l \geq 0.85(30 - m/7) \tag{38}$$

（3） **建立模型**

基于问题二所建立模型，增加决策变量d_t^l表示第l样本的完成指示变量、决策变量w^l表示第l样本的整周完成指标变量、决策变量z_t^l表示第l样本第t天的预测需求以及决策变量$\bar{s}_t^l$表示第l样本第t天在预测需求下生产的 WPCR 实际分配库存。基于问题二模型，结合所列约束条件(28)~(38)式，建立本问题模型：

$$\min \sum_{l}\sum_{t}\sum_{i} CP_i p_{ti}^l + CS_i s_{ti}^l$$

约束条件如式(39)所示，由于式(39)过长考虑排版问题，将其分为式(39-1)和式(39-2)

$$s.t.\begin{cases} x_{t1}^l + s_{t-1,1}^l \geq z_t^l \\ \sum_i TC_i x_{ti}^l \leq TU_t \\ x_{ti}^l \leq p_{ti}^l M \\ s_{ti}^l = s_{t-1,i}^l + x_{ti}^l - u_{ti}^l \\ s_{0i}^l = s_{7i}^k, 1 \leq l,k \leq 30 - n/7 \\ s_{t-1,j}^l \geq u_{tj}^l, 2 \leq j \leq 12 \\ u_{tj}^l = \sum_i RP_{ij} x_{ti}^l, 2 \leq j \leq 12 \\ z_t^l = f(Y^l)_t \\ u_{t1}^l = z_t^l \\ \bar{s}_t^l = \left(\bar{s}_{t-1}^l + x_{ti}^l - Z_t^l\right)d_t^l \\ \bar{s}_0^l = s_{01}^l \end{cases} \tag{39-1}$$

$$s.t.\begin{cases} \sum_t x_{t1}^l \geq s_{01}^l \\ \bar{s}_{t-1}^l + x_{ti}^l - Z_t^l + (1-d_t^l)M \geq 0 \\ \sum_l \sum_t d_t^l \geq 0.95(210-m) \\ \sum_t d_t^l \geq 7w^l \\ \sum_l w^l \geq 0.85\left(30-\frac{m}{7}\right) \\ \bar{s}_t^l, z_t^l, x_{ti}^l, u_{ti}^l, s_{ti}^l \in N \\ w^l, d_t^l, p_{ti}^l \in \{0,1\} \end{cases} \quad (39-2)$$

在式(39)，即式(39-1)和式(39-2)中，如无特殊说明，$1 \leq t \leq 7$，$1 \leq l,k \leq 30-m/7$，$1 \leq i \leq 12$。式（30）种f可以选择不同的预测器的模型，一种选择方案为线性预测器，取$m=7$，令

$$z = f(y) = Ay$$

其中，$A \in \mathbb{R}^{n\times m}$。在文章后续求解中，均采用该方案。

求解得到A和s_{7i}，利用第 30 周数据计算出第 31 周的预测需求，然后令问题 2 中模型每周初始备料量s_{0i}等于所求的优化的每周初始备料量s_{7i}，求解即可得到未来一周的工作计划。

5.4.2 模型的简化

本问题由于所建立的优化预测器参数和预留的组件量的模型过于复杂，在利用 Gurobi 软件编程求解最优策略时，消耗资源过大，难以得到可行解，求解效率低。为了更方便求解出最优策略，考虑对上述模型进行简化，减低其复杂度，增加程序运行效率。这里将参数优化部分模型进行简化，建立两个模型分别优化未来 WPCR 需求预测器和预留的组件量，提高求解效率。

这里我们假设每周预留给下周的组件或 WPCR 数目一定；假设生产单件 WPCR 的成本恒定，该成本由问题 2 的平均成本给出，可视为平均最小成本；因为我们假设每周预留固定数目的组件，所以关键设备工时在一周内可以任意调配，对关键设备的工时限制控制，可以通过控制一周内生产所需 WPCR 的 ABC 三种组件的总工时来完成。设生产单件 WPCR 的平均成本为常量CM。

为计算出未来某周 7 天订单数的预测需求，需要在历史周订单数据中采样。采样方法仍为：z的实际值取Z整周 7 天 WPCR 需求，Y取采样周前m天的历史数据，m为n的整数倍。按照此方法共可采样获得$30-m/7$个样本。

设置决策变量d_t^l表示第l样本的完成指示变量，$d_t^l=1$表示在第l样本中，第t天的生产满足需求，即 WPCR 存量和当天生产量可供给当天 WPCR 需求；否则当$d_t^l=0$，当天不满足需求。决策变量w^l表示第l样本的整周完成指标变量，$w^l=1$表示第l样本整周完成 WPCR 需求，即整周内任意一天都满足 WPCR 需求；否则，第l样本的当周不满足

需求。z_t^l表示第l样本第t天的预测需求。

（1） 优化映射f的参数

$$\min CM\sum_l\sum_t z_t^l \tag{40}$$

$$s.t.\begin{cases} z_t^l = f(Y^l)_t \\ \sum_t z_t^l \sum_i RP_{1i}TC_i \le \sum_t TU_t \\ z_t^l - Z_t^l + (1-d_t^l)M \ge 0 \\ \sum_l\sum_t d_t^l \ge 0.95(210-m) \\ \sum_t d_t^l \ge 7w^l \\ \sum_l w^l \ge 0.85\left(30-\frac{m}{7}\right) \\ z_t^l \in N \\ w^l, d_t^l \in \{0,1\} \end{cases} \tag{41}$$

在式(41)中，如无特殊说明，$1 \le t \le 7$，$1 \le l,k \le 30-m/7$，$1 \le i \le 12$。约束条件分别为：预测函数约束；周总工时约束；WPCR 使用量大于实际需求量时，标记为当天满足需求的约束；95%以上的天数满足需求约束；当周所有天数均满足需求时，整周满足需求约束；85%以上的周次满足约束以及z_t^l、w^l、d_t^l取值约束。

（2） 修改原模型优化预先备料数量

设模型得到的预测值$\bar{Z}_t^l = f(Y^l)_t$

$$\min\sum_l\sum_t\sum_i CP_i p_{ti}^l + CS_i s_{ti}^l$$

约束条件如式(42)所示，由于式(42)过长考虑排版问题，将其分为式(42-1)和式(42-2)

$$s.t.\begin{cases} x_{t1}^l + s_{t-1,1}^l \ge \bar{Z}_t^l \\ \sum_i TC_i x_{ti}^l \le TU_t \\ x_{ti}^l \le p_{ti}^l M \\ s_{ti}^l = s_{t-1,i}^l + x_{ti}^l - u_{ti}^l \\ s_{0i}^l = s_{7i}^k, 1 \le l,k \le 30-n/7 \\ s_{t-1,j}^l \ge u_{tj}^l, 2 \le j \le 12 \\ u_{tj}^l = \sum_i RP_{ij}x_{ti}^l, 2 \le j \le 12 \\ u_{t1}^l = \bar{Z}_t^l \\ \bar{s}_t^l = (\bar{s}_{t-1}^l + x_{ti}^l - Z_t^l)d_t^l \\ \bar{s}_0^l = s_{01}^l \end{cases} \tag{42-1}$$

$$s.t.\begin{cases}\bar{s}_{t-1}^{l}+x_{ti}^{l}-Z_t^l+\left(1-d_t^l\right)M\geq 0\\ \sum_l\sum_t d_t^l\geq 0.95(210-m)\\ \sum_t d_t^l\geq 7w^l\\ \sum_l w^l\geq 0.85\left(30-\frac{m}{7}\right)\\ \bar{s}_t^l,x_{ti}^l,u_{ti}^l,s_{ti}^l\in N\\ w^l,d_t^l,p_{ti}^l\in\{0,1\}\end{cases}\qquad(42-2)$$

在式(42)，即式(42-1)和式(42-2)中，如无特殊说明，式（1）-（17）中$1\leq t\leq 7$，$1\leq l,k\leq 30-m/7$，$1\leq i\leq 12$。相比于原模型，修改了第l样本第t天预测的 WPCR 需求约束式（28）以及 WPCR 使用量等于需求量约束式（31），主要是用$\bar{Z}_t^l$代替原z_t^l；删去了原来预测函数约束式（30）以及生产量大于库存量约束式（34），删去这两个约束条件是因为映射f为固定值，f不再作为优化的目标，同时生产量大于库存量约束式（34）失去意义。

（3） 预测模型

利用预测器f计算所需 WPCR 数量，加上计算得到的预先备料数量s_{0i}。设预测的需求数量为常量Z_t；优化的预先备料数量为常量$S0_i$。利用修改后的问题二模型进行预测，得到结果。修改后的问题二模型为：

$$\min\sum_t\sum_i CP_ip_{ti}+CS_is_{ti}$$

$$s.t.\begin{cases}x_{t1}+s_{t-1,1}\geq Z_t\\ \sum_i TC_ix_{ti}\leq TU_t\\ x_{ti}\leq p_{ti}M\\ s_{ti}=s_{t-1,i}+x_{ti}-u_{ti}\\ s_{0i}=s_{7i}=S0_i\\ s_{t-1,j}\geq u_{tj},2\leq j\leq 12\\ u_{tj}=\sum_i RP_{ij}x_{ti},2\leq j\leq 12\\ u_{t1}=R_t\\ x_{ti},u_{ti},s_{ti}\in N\\ p_{ti}\in\{0,1\}\end{cases}\qquad(43)$$

5.4.3 模型的求解

本问题由于所建立的优化预测器参数和预留的组件量的模型过于复杂，在利用 python 语言编程调用 Gurobi 软件编程求解最优策略时，求解效率低，难以得到可行解。这里将参数优化部分模型进行简化，建立两个模型分别优化未来 WPCR 需求预测器和预留的组件量，提高求解效率。

优化后未来 WPCR 需求预测器参数A（假设为线性预测器）的值如表 8 所示。修改原模型优化预先备料数量后得到的表示每周初始备料量的优化值如表 9 所示。

表 8：优化后未来 WPCR 需求预测器参数 A 取值

维度	1	2	3	4	5	6	7
1	0.6901	-0.4916	-0.0224	-1.3098	0.5815	0.8535	0.8336
2	0.3566	-1.5540	0.6124	-1.9300	1.6566	1.4444	0.5002
3	0.4792	-0.3524	0.1207	-0.5934	0.4605	0.5513	0.4403
4	0.6962	0.2409	0.1175	-0.5045	-0.3604	0.4983	0.5044
5	-0.0254	0.2473	0.5154	0.5713	0.0184	-0.0532	-0.2226
6	-0.1408	0.4010	-0.0931	0.2766	0.5243	-0.1106	0.1450
7	0.2429	-0.3241	0.4136	-0.0343	0.5454	0.2123	0.0891

表 9：每周初始备料量的优化值

组件	WPCR	A	A1	A2	A3	B	B1	B2	C	C1	C2	C3
每周初始备料量优化值	48	0	1500	2000	500	0	688	1336	0	3320	830	4980

由表 9 可知，简化后模型修改原模型优化预先备料数量后得到的 WPCR、A、A1、A2、A3、B、B1、B2、C、C1、C2、C3 共计 12 类组件每周初始备料量的优化值分别为：48、0、1500、2000、500、0、668、1336、0、3320、830、4980 件。

接下来对简化后的模型，进行求解。得到根据历史周订单数据，计算出未来某周 7 天 WPCR 订单数的预测需求如表 10 所示。在最优总成本期望下，制定的一个稳妥的单周生产计划，整理求解结果如表 11 所示。

表 10：未来某周 7 天 WPCR 订单数的预测需求

天	周一	周二	周三	周四	周五	周六	周日
WPCR 需求	35	22	37	44	41	40	40

表 11：问题 4 结果数据

日期	WPCR 组装数量	A 组装数量	B 组装数量	C 组装数量	生产准备费用	库存费用
周一	61	31	437	444	870	49533
周二	89	0	155	260	580	23827
周三	52	0	0	0	420	2032
周四	0	0	177	0	480	5997
周五	0	246	0	308	650	37035
周六	57	0	267	283	730	27485
周日	0	500	0	0	620	34695
总和	259	777	1036	1295	184954	

六、 模型的评价与推广

6.1 模型的优点

1) 针对问题 1-问题 4 建立的混合整型规划模型，目标函数设计合理，约束全面，能够获得全局最优/次优解。
2) 针对问题 3 建立的模型，不失准确性地将非线性约束简化成线性约束，将非线性模型转化为线性模型，使原模型易于求解。
3) 针对问题 4 建立的模型，设计了每周组件备料量一致，未来一周的 WPCR 只与前几周 WPCR 需求量相关等一系列合理假设，不失一般性地简化了模型设计。
4) 针对问题 4 建立的模型的简化求解模型，设计了未来 WPCR 需求量预测器是线性预器，未来一周 WPCR 需求量仅与过去一周的 WPCR 需求量相关等假设，同时将未来 WPCR 需求量预测器参数和每周备料量分成两个模型进行优化，简化了原规模较大的难以得到可行解的混合整型规划模型，大幅提高了求解效率。

6.2 模型的不足与改进方向

1) 针对问题 3 建立的模型，尽管已经转化为线性模型，但在短时间内（1200 秒内）仍然无法获得最优解，只能得到的次优解，后续可以考虑升级硬件设备或优化模型。
2) 针对问题 4 建立的模型，模型假设每周组件备料量一致的合理但简单假设，但实际生产中可能采样更复杂的组件备料量设定方式。
3) 针对问题 4 建立的模型，模型将未来 WPCR 需求量预测器设计为线性函数，并假设未来一周 WPCR 需求量仅与过去一周的 WPCR 需求量相关，但在实际生产中，未来 WPCR 需求量预测器可能是非线性，并且可能与过去好几周的 WPCR 需求量相关，后续考虑改进非线性的 WPCR 需求量预测器。
4) 针对问题 4 建立的模型的简化求解，将预测器参数和每周组件备料量分开求解，虽然提高了求解效率，但牺牲了成本上的最优性，求解的结果在成本上可能只是次优的，在不考虑资源消耗和求解效率，优先保证最优策略时，可改进模型将预测器参数和每周组件备料量组合求解。

七、 参考文献

[1] 姜启源，谢金星，叶俊．数学模型．北京：高等教育出版社，2011：444-446．

[2] 司守奎，孙兆亮．数学建模算法与应用．北京：国防工业出版社，2017：738-743．

[3] 张德强．多级制造车间生产计划与调度集成优化研究[D]．沈阳工业大学，2015．

[4] 张晓东，严洪森．多级车间生产计划和调度的集成优化[J]．机械工程学报，2005(09)：98-105．

[5] 张萃珠，李志滔，余秋萍，赵莹，何演铭，龙建宇，洪颖．基于混合整数规划的 P 型玩具生产线平衡优化[J]．机电工程技术，2022，51(05)：198-203．

[6] 程凤，霍佳震．基于混合整数规划的冶炼型材生产组织优化[J]．上海管理科学，2011，33(06)：40-42．

[7] Optimization LLC. Gurobi optimizer reference manual[J]. 2022.

八、 附录

8.1 使用的软件

相关软件使用说明
1. 文字处理软件： Word 2019。 2. 编程语言：Python 3.8。 3. 编程软件：Pycharm 2021。 3. 线性规划求解软件：Gurobi 9.5.2。 4. 画图软件：Xmind。 5. 公式编辑软件：office 公式编辑器。

8.2 源程序代码说明

源程序代码说明
代码基于 Python 编程语言，主要使用 Gurobi 优化求解器对问题模型进行优化求解，问题 1 到问题 4 的代码分别为 question1.py - question4*.py，各代码中模型所需参数/参数读取代码已经写好，直接运行即可得到结果。
问题 1 运行 question1.py 可以得到运行结果 question1.csv 文件， question1.csv 中数值代表对应天数和对应组件的生产数，对应组件的总生产数，对应天数的对应生产/储存成本和总成本。如：Day1 行，WPCR 列对应第一天 WPCR 生产量。
问题 2 运行 question2.py 可以得到运行结果 question2.csv 文件， question2.csv 中数值含义于问题 1 中 *.csv 文件一致。 verify.py 程序为问题一、二模型验证程序。
问题 3 运行 question3.py 可以得到运行结果 question3.csv 文件， question3.csv 中第 2 行数值表示总生产成本，第 3 行表示检查日期，后续出现的表格中数值含义与问题 1/2 中 *.csv 文件一致。 由于问题 3 规模较大，Gurobi 难以在短时间内求得最优解，所以 question3.py 设定最大求解时限参数为 1200s （见 question3.py 第 28 行），解的质量与求解时限呈正相关，求解时限越大，消耗时间更多，解的质量更高。可以调整该参数在时间和解的质量中做权衡。 这部分代码需要 datas.xlsx 作为参数输入文件，该文件内包含 30 天内 WPCR 需求量文件，保证 datas.xlsx 在根目录下即可。
问题 4 问题 4 分 3 个模型依次进行求解，存在 3 个代码文件：question4-1.py, question4-2.py, question4-3.py，按照顺序依次运行 question4-1.py - question4-3.py 可依次得到 question4-1.csv -

question4-3.csv。

这部分代码需要 datas.xlsx 作为参数输入文件，该文件内包含历史 30 天内 WPCR 需求量文件，保证 datas.xlsx 在根目录下即可。

模型 1
运行 question4-1.py 可以得到运行结果 question4-1.csv 文件，该代码用于求解优化预测器参数的模型。question4-1.csv 中结果分为 3 个部分，第 1 部分为优化参数 A 的值，一个 7 X 7 矩阵；第 2 部分为各样本对未来一周各天的 WPCR 需求量预测值，D1-D7 表示周一到周日；第 3 部分为每天/每周是否完成需求的指示变量值。

模型 2
运行 question4-2.py 可以得到运行结果 question4-2.csv 文件，该代码用于求解优化每周初始备料量的模型。question4-2.csv 中结果分为 3 个部分，第 1 部分为第 3 行，表示每周初始备料量的优化值，共 12 个值，对应 12 个组件（包括 WPCR）；第 2 部分为各样本对未来一周各天的 WPCR 需求量预测值，D1-D7 表示周一到周日；第 3 部分为每天/每周是否完成需求的指示变量值。

这部分代码需要 question4-1.csv 作为参数输入文件，提供优化的预测器参数。

由于该模型规模较大，Gurobi 难以在短时间内求得最优解，所以 question4-2.py 设定最大求解时限参数为 1200s （见 question4-2.py 第 32 行），解的质量与求解时限呈正相关，求解时限越大，消耗时间更多，解的质量更高。可以调整该参数在时间和解的质量中做权衡。

模型 3
运行 question4-3.py 可以得到运行结果 question4-3.csv 文件，该代码用于预测未来一周 WPCR 需求量和安排生产计划。question4-3.csv 中结果分为 2 个部分，第 1 部分为第 3 行，表示预测的未来一周 WPCR 需求量，对应未来一周周一到周日的需求量；第 2 部分为未来一周的生产计划，数值含义与问题 1/2 中 *.csv 文件一致。

这部分代码需要 question4-1.csv 作为参数输入文件，提供优化的预测器参数；需要 question4-2.csv 作为参数输入文件，提供优化的每周初始备料量。

8.3 相关源程序代码

question1：第一问模型求解代码（python 编写，调用 Gurobi 软件）

```
import numpy as np
import gurobipy as gp
from gurobipy import GRB
global MIPM

MIPM = 1e6    # big M method
```

```
class WPCRP():
    def init(self, name=""):
        self.modelname = name
        self.T = 0
        self.N = 0
        self.CP = None
        self.CS = None

    def solve(self, paramSet=None):
        global MIPM

        T, N, R, TU, TC, CP, CS, RP = paramSet
        self.T,self.N,self.CP,self.CS = T,N,CP,CS
        # declare model
        m = gp.Model(self.modelname)

        # Time Limited
        m.setParam(GRB.Param.TimeLimit, 240.0)
        # Min Gap
        # m.setParam(GRB.Param.MIPGap, 0.05)

        # define the decesion variable
        x = m.addMVar(shape=(T, N),   vtype=GRB.INTEGER,   name="x")
        u = m.addMVar(shape=(T, N),   vtype=GRB.INTEGER,   name="u")
        s = m.addMVar(shape=(T+1, N),   vtype=GRB.INTEGER,   name="s")
        p = m.addMVar(shape=(T, N),   vtype=GRB.BINARY,   name="p")

        for t in range(T):
            m.addConstr(x[t][0]+s[t][0] >= R[t])   # eq.2
            m.addConstr(x[t][1]TC[1] + x[t][5]TC[5] +
                        x[t][8]TC[8] = u[t][j]) # eq.8
                        m.addConstr(u[t][j] == RP[i][j]x[t][i]) # eq.9
        m.setobjective(gp.quicksum(CP[i]    p[t][i]  +  CS[i]    s[t][i]    for  t  in  range(T)  for  i  in
range(N)))

        # save the model
        # m.write('model.lp')

        # compute IIS
        # m.computeIIS()
        # m.write("model.ilp")

        # solve
```

```
        m.optimize()

        # print solution
        print("Solve Status", m.Status)
        self.Status = m.Status
        if self.Status !=2:
            print("NO optimal solution found.")


        self.Sol = x.x,u.x,s.x,p.x

    def GetSol(self):
        MIPFloat = 0.9999
        solx, , sols, solp = self.Sol
        X=np.zeros((self.T,4),dtype=int)
        C=np.zeros((self.T,2),dtype=int)
        partList=[0,1,5,8]
        for t in range(self.T):
            count=0
            for i in partList:
                X[t][count] = round(solx[t][i])
                count+=1
            for i in range(N):
                if solp[t][i]>MIPFloat:
                    C[t][0] += self.CP[i]
                C[t][1] += round(sols[t][i])    self.CS[i]
        return X,C,X.sum(0),C.sum()


    def write(self, filename):
        X,C,XS,CS = self.GetSol()
        split = ' '
        with open(filename, "w", encoding='utf-8') as f:
            f.write("# Solution for model " + self.modelname + "n")
            f.write(split.join(['Data', 'WPCR', 'A', 'B', 'C', 'ReadyCost', 'StoreCost']) + 'n')
            for t in range(self.T):
                f.write('Day'+str(t+1)+split)
                f.write(split.join(str(x) for x in X[t]))
                f.write(split)
                f.write(split.join(str(x) for x in C[t]))
                f.write('n')
            f.write('Sumt')
            f.write(split.join(str(x) for x in XS))
            f.write(split+str(CS))
```

```
if name=='main':
    N=12
    T=7
    R=np.array([39, 36, 38, 40, 37, 33, 40])
    TU=np.array([4500, 2500, 2750, 2100, 2500, 2750, 1500])
    TC= np.array([0, 3, 0, 0 , 0, 5, 0, 0, 5, 0, 0, 0])
    CP=np.array([240, 120, 40, 60, 50, 160, 80, 100, 180, 60, 40, 70])
    CS=np.array([5, 2, 5, 3, 6, 1.5, 4, 5, 1.7, 3, 2, 3])
    RP=np.zeros((N,N))

RP[0][1],RP[0][5],RP[0][8],RP[1][2],RP[1][3],RP[1][4],RP[5][6],RP[5][7],RP[8][9],RP[8][10],RP[8][11]=3,4,5,6,8,2,2,4,8,2,12


    wpcrp=WPCRP("WPCRP 1")
    paramSet = T, N, R, TU, TC, CP, CS, RP
    wpcrp.solve(paramSet)
    print(wpcrp.Sol)
    print(wpcrp.GetSol())
    wpcrp.write("question1.csv")
```

question2：第二问模型求解代码（python 编写、Gurobi 软件）

```
import numpy as np
import gurobipy as gp
from gurobipy import GRB

global MIPM

MIPM = 1e6    # big M method


class WPCRP():
    def init(self, name=""):
        self.modelname = name
        self.T = 0
        self.N = 0
        self.CP = None
        self.CS = None

    def solve(self, paramSet=None):
```

```
        global MIPM

        T, N, R, TU, TC, CP, CS, RP = paramSet
        self.T, self.N, self.CP, self.CS = T, N, CP, CS
        # declare model
        m = gp.Model(self.modelname)

        # Time Limited
        m.setParam(GRB.Param.TimeLimit, 240.0)
        # Min Gap
        # m.setParam(GRB.Param.MIPGap, 0.05)

        # define the decesion variable
        x = m.addMVar(shape=(T, N), vtype=GRB.INTEGER, name="x")
        u = m.addMVar(shape=(T, N), vtype=GRB.INTEGER, name="u")
        s = m.addMVar(shape=(T + 1, N), vtype=GRB.INTEGER, name="s")
        p = m.addMVar(shape=(T, N), vtype=GRB.BINARY, name="p")

        for t in range(T):
            m.addConstr(x[t][0] + s[t][0] >= R[t])    # eq.2
            m.addConstr(x[t][1]    TC[1] + x[t][5]    TC[5] +
                                x[t][8]    TC[8] = u[t][j])    # eq.7
                                m.addConstr(u[t][j] == RP[i][j]    x[t][i])    # eq.8
        m.setobjective(gp.quicksum(CP[i]    p[t][i] + CS[i]    s[t][i] for t in range(T) for i in range(N)))

        # save the model
        # m.write('model.lp')

        # compute IIS
        # m.computeIIS()
        # m.write("model.ilp")

        # solve
        m.optimize()

        # print solution
        print("Solve Status", m.Status)
        self.Status = m.Status
        if self.Status != 2:
            print("NO optimal solution found.")

        self.Sol = x.x, u.x, s.x, p.x

    def GetSol(self):
```

```
            MIPFloat = 0.9999
            solx, , sols, solp = self.Sol
            X = np.zeros((self.T, 4), dtype=int)
            C = np.zeros((self.T, 2), dtype=int)
            partList = [0, 1, 5, 8]
            for t in range(self.T):
                count = 0
                for i in partList:
                    X[t][count] = round(solx[t][i])
                    count += 1
                for i in range(N):
                    if solp[t][i] > MIPFloat:
                        C[t][0] += self.CP[i]
                    C[t][1] += round(sols[t][i])    self.CS[i]
            return X, C, X.sum(0), C.sum()

        def write(self, filename):
            X, C, XS, CS = self.GetSol()
            split = ' '
            with open(filename, "w", encoding='utf-8') as f:
                f.write("# Solution for model " + self.modelname + "n")
                f.write(split.join(['Data', 'WPCR', 'A', 'B', 'C', 'ReadyCost', 'StoreCost']) + 'n')
                for t in range(self.T):
                    f.write('Day' + str(t + 1) + split)
                    f.write(split.join(str(x) for x in X[t]))
                    f.write(split)
                    f.write(split.join(str(x) for x in C[t]))
                    f.write('n')
                f.write('Sumt')
                f.write(split.join(str(x) for x in XS))
                f.write(split + str(CS))

if name=='main':
    N = 12
    T = 7
    R = np.array([39, 36, 38, 40, 37, 33, 40])
    TU = np.array([4500, 2500, 2750, 2100, 2500, 2750, 1500])
    TC = np.array([0, 3, 0, 0, 0, 5, 0, 0, 5, 0, 0, 0])
    CP = np.array([240, 120, 40, 60, 50, 160, 80, 100, 180, 60, 40, 70])
    CS = np.array([5, 2, 5, 3, 6, 1.5, 4, 5, 1.7, 3, 2, 3])
    RP = np.zeros((N, N))
    RP[0][1], RP[0][5], RP[0][8], RP[1][2], RP[1][3], RP[1][4], RP[5][6], RP[5][7], RP[8][9],
RP[8][10], RP[8][
        11] = 3, 4, 5, 6, 8, 2, 2, 4, 8, 2, 12
```

```
wpcrp = WPCRP("WPCRP 2")
paramSet = T, N, R, TU, TC, CP, CS, RP
wpcrp.solve(paramSet)
print(wpcrp.GetSol())
wpcrp.write("question2.csv")
```

verify：第一、二问模型验证代码（python）

```
import pandas as pd
import numpy as np

source = pd.readcsv('question2.csv',sep = 's')

data = source.iloc[1:8,:]
data = np.array(data)      #  （76）  一周的 WPCR A B C ReadyCost StoreCost
data = data.astype(int)

print(data)

flag = [1,1,1]    # 依次为工时需求是否满足、组装 WPCR 的 ABC 工件需求是否满足、每日 WPCR 需求数是否满足
Timeg = [4500,2500,2750,2100,2500,2750,1500]    # 每日总工时限制
Timeg = np.asarray(Timeg)
Timet = np.zeros((7,), dtype = int)                  # 每日实际消耗总工时

for i in range(len(data)):
    Timet[i] = 3data[i][1] + 5data[i][2] + 5data[i][3]
    if(Timet[i] > Timeg[i]):
        flag[0] = 0

print(Timet)

Workpiece = np.zeros((7,3), dtype = int)          #   每日结束剩余 ABC 工件
Workpiece[0][0] = data[0][1] - 3data[0][0]
Workpiece[0][1] = data[0][2] - 4data[0][0]
Workpiece[0][2] = data[0][3] - 5data[0][0]

WPCR = np.zeros((7,), dtype = int)                #   每日结束剩余 WPCR
WPCRg = [39,36,38,40,37,33,40]
WPCRg =    np.asarray(WPCRg)
WPCR[0] = data[0][0] - WPCRg[0]
```

```
for i in range(1,len(data)):
    # 更新每日剩余 ABC 工件
    Workpiece[i][0] = Workpiece[i - 1][0] + data[i][1] - 3    data[i][0]
    Workpiece[i][1] = Workpiece[i - 1][1] + data[i][2] - 4    data[i][0]
    Workpiece[i][2] = Workpiece[i - 1][2] + data[i][3] - 5    data[i][0]
    # 更新每日剩余 WPCR
    WPCR[i] = WPCR[i - 1] + data[i][0] - WPCRg[i]

print(Workpiece)
if(np.min(Workpiece) < 0):
    flag[1] = 0

print(WPCR)
if(np.min(WPCR) < 0):
    flag[2] = 0
print(flag)
```

question3：第三问模型求解代码（python 编写、Gurobi 软件）

```
import pandas as pd
import numpy as np
import gurobipy as gp
from gurobipy import GRB

global MIPM

MIPM = 1e4    # big M method


class WPCRP():
    def init(self, name=""):
        self.modelname = name
        self.T = 0
        self.N = 0
        self.CP = None
        self.CS = None

    def solve(self, paramSet=None):
        global MIPM

        S, DS, DT, T, N, DTU, R, TU, TC, CP, CS, RP = paramSet
        self.S, self.T, self.N, self.CP, self.CS = S, T, N, CP, CS
        # declare model
```

```
        m = gp.Model(self.modelname)

        # Time Limited
        m.setParam(GRB.Param.TimeLimit, 1200.0)
        # Min Gap
        # m.setParam(GRB.Param.MIPGap, 0.05)

        # define the decesion variable
        x = m.addMVar(shape=(T, N), vtype=GRB.INTEGER, name="x")
        u = m.addMVar(shape=(T, N), vtype=GRB.INTEGER, name="u")
        s = m.addMVar(shape=(T + 1, N), vtype=GRB.INTEGER, name="s")
        p = m.addMVar(shape=(T, N), vtype=GRB.BINARY, name="p")
        c = m.addMVar(shape=(T), vtype=GRB.BINARY, name="c")
        v = m.addMVar(shape=(T), vtype=GRB.CONTINUOUS, name='v')

        m.addConstr(c.sum() == S)    # eq. 11

        for t in range(T):
            m.addConstr(x[t][0] + s[t][0] >= R[t])    # eq.2
            m.addConstr(x[t][1] * TC[1] + x[t][5] * TC[5] +
                        x[t][8]    TC[8] <= v[t] )    # eq.3

            m.addConstr(u[t][0] == R[t])    # eq.9

        for i in range(N):
            m.addConstr(s[0][i] == s[7][i])    # eq.6

        for t in range(T):
            for i in range(N):
                m.addConstr(x[t][i] = u[t][j])    # eq.7
                        m.addConstr(u[t][j] == RP[i][j]    x[t][i])    # eq.8

        for t in range(211 - DS):
            m.addConstr(c[t:t + DS].sum()    MIPFloat:
                CD[count] = t + 1
                count += 1
        return X, C, X.sum(0), C.sum(), CD

    def write(self, filename):
        X, C, XS, CS, CD = self.GetSol()
        split = ' '
        with open(filename, "w", encoding='utf-8') as f:
            f.write("# Solution for model " + self.modelname + "n")
            f.write('# Sum Cost: ' + str(CS) + 'n')
```

```
                f.write('# Checking Date: ')
                f.write(split.join(str(x) for x in CD) + 'n')
                f.write(split.join(['Data', 'WPCR', 'A', 'B', 'C', 'ReadyCost', 'StoreCost']) + 'n')
                for t in range(self.T):
                    f.write('Week' + str(int(t / 7)) + '-Day' + str(t % 7 + 1) + split)
                    f.write(split.join(str(x) for x in X[t]))
                    f.write(split)
                    f.write(split.join(str(x) for x in C[t]))
                    f.write('n')
                f.write('Sumt')
                f.write(split.join(str(x) for x in XS))
                f.write(split + str(CS))


if name == 'main':
    data = pd.readexcel('datas.xlsx')

    data = data.iloc[:, -7:]
    data30 = np.array(data)   #    (307)
    data210 = data30.reshape(-1)

    S = 7
    DS = 6
    DT = 5
    N = 12
    T = 210
    NWeek = int(T / 7)
    DTU = [0.1, 0.08, 0.06, 0.04, 0.02]
    R = data210.astype(int)
    TU = np.array([4500, 2500, 2750, 2100, 2500, 2750, 1500]   NWeek)
    TC = np.array([0, 3, 0, 0, 0, 5, 0, 0, 5, 0, 0, 0])
    CP = np.array([240, 120, 40, 60, 50, 160, 80, 100, 180, 60, 40, 70])
    CS = np.array([5, 2, 5, 3, 6, 1.5, 4, 5, 1.7, 3, 2, 3])
    RP = np.zeros((N, N))
    RP[0][1], RP[0][5], RP[0][8], RP[1][2], RP[1][3], RP[1][4], RP[5][6], RP[5][7], RP[8][9],
RP[8][10], RP[8][
        11] = 3, 4, 5, 6, 8, 2, 2, 4, 8, 2, 12

    wpcrp = WPCRP("WPCRP 3")
    paramSet = S, DS, DT, T, N, DTU, R, TU, TC, CP, CS, RP
    wpcrp.solve(paramSet)

    # print(wpcrp.GetSol())
    wpcrp.write("question3e.csv")
```

question4-1：求解优化预测器参数的模型（python 编写、Gurobi 软件）

```
import pandas as pd
import numpy as np
import gurobipy as gp
from gurobipy import GRB

global MIPM

MIPM = 1e4    # big M method


class WPCRP():
    def init(self, name=""):
        self.modelname = name
        self.T = 0
        self.N = 0
        self.CP = None
        self.CS = None

    def solve(self, paramSet=None):
        global MIPM

        T, N, R, TU, TC, CM,RP = paramSet
        Y,Z=R
        NY,NZ=Y.shape[-1],Z.shape[-1],
        NS =len(Y)
        CT=(RP[0]TC).sum()
        self.NS, self.T, self.N, self.CP, self.CS = NS, T, N, CP, CS
        # declare model
        m = gp.Model(self.modelname)

        # Time Limited
        # m.setParam(GRB.Param.TimeLimit, 360.0)
        # Min Gap
        # m.setParam(GRB.Param.MIPGap, 0.05)

        d = m.addMVar(shape=(NS, T), vtype=GRB.BINARY, name="d")
        w = m.addMVar(shape=(NS), vtype=GRB.BINARY, name="w")
        z = m.addMVar(shape=(NS, T), vtype=GRB.CONTINUOUS, name="z")
        a = m.addMVar(shape=(NZ, NY),lb=-10,ub=10, vtype=GRB.CONTINUOUS, name="a")

        for l in range(NS):
```

```
            # m.addConstr(z[l] == a@Y[l]) # eq.9

            m.addConstr(d[l].sum()>=7w[l]) # eq.7
            m.addConstr(CTz[l].sum()= 0 )    # eq.4
            for n in range(NZ):
                m.addConstr(z[l][n] == gp.quicksum(a[n][r]Y[l][r] for r in range(NY))) # eq.2


        m.addConstr(d.sum()>=0.95NST) # eq.5
        m.addConstr(w.sum()>=0.85NS) # eq.7

        m.setobjective(CMz.sum())

        # save the model
        # m.write('model.lp')

        # compute IIS
        # m.computeIIS()
        # m.write("model.ilp")

        # solve
        m.optimize()

        # print solution
        print("Solve Status", m.Status)
        self.Status = m.Status
        if self.Status != 2:
            print("NO optimal solution found.")
        m.write('output4.sol')
        self.Sol = d.x,w.x,z.x,a.x

    def GetSol(self):

        sold, solw, solz, sola = self.Sol

        Z = np.zeros((self.NS,self.T), dtype=int)
        D = np.zeros((self.NS,self.T), dtype=int)
        W = np.zeros(self.NS, dtype=int)
        A = sola

        for l in range(self.NS):
            W[l] = round(solw[l])
            for t in range(self.T):
                Z[l][t] = round(solz[l][t])
```

```
                    D[l][t] =round(sold[l][t])


        return Z,D,W,A

    def write(self, filename):
        Z,D,W,A = self.GetSol()
        split = ' '
        with open(filename, "w", encoding='utf-8') as f:
            f.write("# Solution for model " + self.modelname + "n")
            f.write("# A:n")
            for i in range(A.shape[0]):
                f.write(split.join(str(x) for x in A[i])+'n')
            f.write(split.join(['Sample', 'D1', 'D2', 'D3', 'D4', 'D5', 'D6','D7']) + 'n')
            for l in range(self.NS):
                f.write(str(l + 1)+'Z' + split)
                f.write(split.join(str(x) for x in Z[l])+'n')

            f.write(split.join(['Sample', 'D1', 'D2', 'D3', 'D4', 'D5', 'D6','D7','Week']) + 'n')
            for l in range(self.NS):
                f.write(str(l + 1) + split)
                f.write(split.join(str(x) for x in D[l]))
                f.write(split+ str(W[l]))
                f.write('n')

def sample(data,m,n):
    d1,d2=data.shape
    if m%d2!=0 or n%d2!=0:
        print("m and n must be the integral multiple of ",d2,".")
    if n/d2 !=1:
        print("n must be ",d2,".")
    m,n=int(m/d2),1
    Y=np.zeros((d1-m,md2))
    Z=np.zeros((d1-m,d2))
    count = 0
    for i in range(m,d1):
        Y[count] = data[i-m:i].reshape(-1)
        Z[count] = data[i]
        count+=1
    return Y,Z



if name=='main':
```

```
    data = pd.readexcel('datas.xlsx')

    data = data.iloc[:,-7:]
    data30 = np.array(data)      #  (307)
    YZ= sample(data30,7,7)

    N = 12
    T = 7
    CM = 674
    R = YZ
    TU = np.array([4500, 2500, 2750, 2100, 2500, 2750, 1500])
    TC = np.array([0, 3, 0, 0, 0, 5, 0, 0, 5, 0, 0, 0])
    CP = np.array([240, 120, 40, 60, 50, 160, 80, 100, 180, 60, 40, 70])
    CS = np.array([5, 2, 5, 3, 6, 1.5, 4, 5, 1.7, 3, 2, 3])
    RP = np.zeros((N, N))
    RP[0][1], RP[0][5], RP[0][8], RP[1][2], RP[1][3], RP[1][4], RP[5][6], RP[5][7], RP[8][9],
RP[8][10], RP[8][
        11] = 3, 4, 5, 6, 8, 2, 2, 4, 8, 2, 12

    wpcrp = WPCRP("WPCRP 4-1")
    paramSet =   T, N, R, TU, TC, CM, RP
    wpcrp.solve(paramSet)
    # print(wpcrp.GetSol())
    wpcrp.write("question4-1.csv")
```

question4-2：求解优化每周初始备料量的模型（python 编写、Gurobi 软件）

```
import pandas as pd
import numpy as np
import gurobipy as gp
from gurobipy import GRB

global MIPM

MIPM = 1e4   # big M method


class WPCRP():
    def init(self, name=""):
        self.modelname = name
        self.T = 0
        self.N = 0
        self.CP = None
```

```
        self.CS = None

    def solve(self, paramSet=None):
        global MIPM

        T, N, R, TU, TC, CP, CS, RP,A = paramSet
        Y,Z=R
        Z = np.matmul(Y, np.transpose(A)).astype(int)
        NY,NZ=Y.shape[-1],Z.shape[-1],
        NS =len(Y)
        self.NS, self.T, self.N, self.CP, self.CS = NS, T, N, CP, CS
        # declare model
        m = gp.Model(self.modelname)

        # Time Limited
        m.setParam(GRB.Param.TimeLimit, 1200.0)
        # Min Gap
        # m.setParam(GRB.Param.MIPGap, 0.05)
        # Foucus
        # m.setParam('MIPFocus', 1)
        # # Heuristics
        # m.setParam('Heuristics',0.5)

        # m.setParam('NonConvex', 2)

        # define the decesion variable
        x = m.addMVar(shape=(NS, T, N), vtype=GRB.INTEGER, name="x")
        u = m.addMVar(shape=(NS, T, N), vtype=GRB.INTEGER, name="u")
        s = m.addMVar(shape=(NS, T + 1, N), vtype=GRB.INTEGER, name="s")
        p = m.addMVar(shape=(NS, T, N), vtype=GRB.BINARY, name="p")
        d = m.addMVar(shape=(NS, T), vtype=GRB.BINARY, name="d")
        w = m.addMVar(shape=(NS), vtype=GRB.BINARY, name="w")
        # z = m.addMVar(shape=(NS, T), vtype=GRB.CONTINUOUS, name="z")
        s = m.addMVar(shape=(NS, T + 1), vtype=GRB.INTEGER, name="s")

        for l in range(NS):

            m.addConstr(d[l].sum()>=7w[l]) # eq.14

            m.addConstr(s[l][0] == s[l][0][0]) # eq.11


            for t in range(T):
                m.addConstr(x[l][t][0] + s[l][t][0] >= Z[l][t])    # eq.2
```

```
                    m.addConstr(x[l][t][1]    TC[1] + x[l][t][5]    TC[5] +
                                x[l][t][8]    TC[8] = 0 )    # eq.12

            for k in range(NS):
                for i in range(N):
                    m.addConstr(s[l][0][i] == s[k][7][i])    # eq.6

            for t in range(T):
                for i in range(N):
                    m.addConstr(x[l][t][i]    0:
                                m.addConstr(s[l][t][j] >= u[l][t][j])    # eq.7
                                m.addConstr(u[l][t][j] == RP[i][j]    x[l][t][i])    # eq.8


        m.addConstr(d.sum()>=0.95NST) # eq.13
        m.addConstr(w.sum()>=0.85NS) # eq.15

        m.setobjective(gp.quicksum(CP[i]    p[l][t][i] + CS[i]    s[l][t][i] for l in range(NS) for t in
range(T) for i in range(N)))

        # save the model
        # m.write('model.lp')

        # compute IIS
        # m.computeIIS()
        # m.write("model.ilp")

        # solve
        m.optimize()

        # print solution
        print("Solve Status", m.Status)
        self.Status = m.Status
        if self.Status != 2:
            print("NO optimal solution found.")
        m.write('output4.sol')
        self.Sol = x.x, u.x, s.x, p.x,d.x,w.x
    def GetSol(self):
        MIPFloat = 0.9999
        solx, solu, sols, solp, sold, solw = self.Sol
        X = np.zeros((self.NS,self.T), dtype=int)
        U = np.zeros((self.NS,self.T), dtype=int)
        S = np.zeros(self.N, dtype=int)
        SS= np.zeros((self.NS,self.T), dtype=int)
```

```
            Z = np.zeros((self.NS,self.T), dtype=int)
            D = np.zeros((self.NS,self.T), dtype=int)
            W = np.zeros(self.NS, dtype=int)



            for i in range(self.N):
                S[i] = sols[0][0][i]
            for l in range(self.NS):
                W[l] = round(solw[l])
                for t in range(self.T):
                    X[l][t] = round(solx[l][t][0])
                    U[l][t] = round(solu[l][t][0])
                    D[l][t] =round(sold[l][t])
                    SS[l][t] = round(sols[l][t][0])

            return X,U, S,D,W,SS

        def write(self, filename):
            X,U,S,D,W,SS = self.GetSol()
            split = ' '
            with open(filename, "w", encoding='utf-8') as f:
                f.write("# Solution for model " + self.modelname + "n")
                f.write("# Store Number:n")
                f.write(split.join(str(x) for x in S)+'n')
                f.write(split.join(['Sample', 'D1', 'D2', 'D3', 'D4', 'D5', 'D6','D7']) + 'n')
                for l in range(self.NS):
                    f.write(str(l + 1)+'X' + split)
                    f.write(split.join(str(x) for x in X[l])+'n')
                    f.write(str(l + 1)+'S' + split)
                    f.write(split.join(str(x) for x in SS[l])+'n')
                    f.write(str(l + 1)+'U' + split)
                    f.write(split.join(str(x) for x in U[l])+'n')



                f.write(split.join(['Sample', 'D1', 'D2', 'D3', 'D4', 'D5', 'D6','D7','Week']) + 'n')
                for l in range(self.NS):
                    f.write(str(l + 1) + split)
                    f.write(split.join(str(x) for x in D[l]))
                    f.write(split+ str(W[l]))
                    f.write('n')

def sample(data,m,n):
        d1,d2=data.shape
        if m%d2!=0 or n%d2!=0:
```

```
            print("m and n must be the integral multiple of ",d2,".")
        if n/d2 !=1:
            print("n must be ",d2,".")
        m,n=int(m/d2),1
        Y=np.zeros((d1-m,md2))
        Z=np.zeros((d1-m,d2))
        count = 0
        for i in range(m,d1):
            Y[count] = data[i-m:i].reshape(-1)
            Z[count] = data[i]
            count+=1
        return Y,Z

def getA(filename):
        data = pd.readcsv(filename)
        data = data.iloc[1:8, :]
        list1 =[]
        for i in range(len(data)):
            list = data.iloc[i][0].strip().split(' ')
            list1.append(list)
        datanp = np.array(list1)
        datanp = datanp.reshape(7,-1).astype(float)
        return datanp


if name=='main':
        data = pd.readexcel('datas.xlsx')

        data = data.iloc[:,-7:]
        data30 = np.array(data)       #    (307)
        YZ= sample(data30,7,7)
        # YZ= YZ[0][:19],YZ[1][:19]
        # data210 = data30.reshape(-1)

        # NS = 29 # the number of samples
        N = 12
        T = 7
        R = YZ #np.array([39, 36, 38, 40, 37, 33, 40])
        TU = np.array([4500, 2500, 2750, 2100, 2500, 2750, 1500])
        TC = np.array([0, 3, 0, 0, 0, 5, 0, 0, 5, 0, 0, 0])
        CP = np.array([240, 120, 40, 60, 50, 160, 80, 100, 180, 60, 40, 70])
        CS = np.array([5, 2, 5, 3, 6, 1.5, 4, 5, 1.7, 3, 2, 3])
        RP = np.zeros((N, N))
```

```
    RP[0][1], RP[0][5], RP[0][8], RP[1][2], RP[1][3], RP[1][4], RP[5][6], RP[5][7], RP[8][9], RP[8][10], RP[8][
        11] = 3, 4, 5, 6, 8, 2, 2, 4, 8, 2, 12

    A = getA('question4-1.csv')
    wpcrp = WPCRP("WPCRP 4-2")
    paramSet =  T, N, R, TU, TC, CP, CS, RP,A
    wpcrp.solve(paramSet)
    # print(wpcrp.GetSol())
    wpcrp.write("question4-2.csv")
```

question4-3：预测未来一周 WPCR 需求量和安排生产计划（python、Gurobi）

```
import numpy as np
import pandas as pd
import gurobipy as gp
from gurobipy import GRB

global MIPM

MIPM = 1e4    # big M method


class WPCRP():
    def init(self, name=""):
        self.modelname = name
        self.T = 0
        self.N = 0
        self.CP = None
        self.CS = None

    def solve(self, paramSet=None):
        global MIPM

        T, N, R, TU, TC, CP, CS, RP, SOI = paramSet
        self.T, self.N, self.R, self.CP, self.CS = T, N, R, CP, CS
        # declare model
        m = gp.Model(self.modelname)

        # Time Limited
        # m.setParam(GRB.Param.TimeLimit, 360.0)
        # Min Gap
        # m.setParam(GRB.Param.MIPGap, 0.05)
```

```
        # define the decesion variable
        x = m.addMVar(shape=(T, N), vtype=GRB.INTEGER, name="x")
        u = m.addMVar(shape=(T, N), vtype=GRB.INTEGER, name="u")
        s = m.addMVar(shape=(T + 1, N), vtype=GRB.INTEGER, name="s")
        p = m.addMVar(shape=(T, N), vtype=GRB.BINARY, name="p")

        for t in range(T):
            m.addConstr(x[t][0] + s[t][0] >= R[t])    # eq.2
            m.addConstr(x[t][1]    TC[1] + x[t][5]    TC[5] +
                        x[t][8]    TC[8] = u[t][j])    # eq.7
                        m.addConstr(u[t][j] == RP[i][j]    x[t][i])    # eq.8
        m.setobjective(gp.quicksum(CP[i]    p[t][i] + CS[i]    s[t][i] for t in range(T) for i in range(N)))

        # save the model
        # m.write('model.lp')

        # compute IIS
        # m.computeIIS()
        # m.write("model.ilp")

        # solve
        m.optimize()

        # print solution
        print("Solve Status", m.Status)
        self.Status = m.Status
        if self.Status != 2:
            print("NO optimal solution found.")

        self.Sol = x.x, u.x, s.x, p.x

    def GetSol(self):
        MIPFloat = 0.9999
        solx, , sols, solp = self.Sol
        X = np.zeros((self.T, 4), dtype=int)
        C = np.zeros((self.T, 2), dtype=int)
        partList = [0, 1, 5, 8]
        for t in range(self.T):
            count = 0
            for i in partList:
                X[t][count] = round(solx[t][i])
                count += 1
            for i in range(N):
```

```
                    if solp[t][i] > MIPFloat:
                        C[t][0] += self.CP[i]
                    C[t][1] += round(sols[t][i])    self.CS[i]
        return X, C, X.sum(0), C.sum()

    def write(self, filename):
        X, C, XS, CS = self.GetSol()
        split = ' '
        with open(filename, "w", encoding='utf-8') as f:
            f.write("# Solution for model " + self.modelname + "n")
            f.write("# Predicted Requirement:n")
            f.write(split.join(str(x) for x in self.R)+'n')
            f.write(split.join(['Data', 'WPCR', 'A', 'B', 'C', 'ReadyCost', 'StoreCost']) + 'n')
            for t in range(self.T):
                f.write('Day' + str(t + 1) + split)
                f.write(split.join(str(x) for x in X[t]))
                f.write(split)
                f.write(split.join(str(x) for x in C[t]))
                f.write('n')
            f.write('Sumt')
            f.write(split.join(str(x) for x in XS))
            f.write(split + str(CS))

def getA(filename):
    data = pd.readcsv(filename)
    data = data.iloc[1:8, :]
    list1 =[]
    for i in range(len(data)):
        list = data.iloc[i][0].strip().split(' ')
        list1.append(list)
    datanp = np.array(list1)
    datanp = datanp.reshape(7,-1).astype(float)
    return datanp

def getSOI(filename):
    data = pd.readcsv(filename)
    data = data.iloc[1:2, :]
    list1 =[]
    for i in range(len(data)):
        list = data.iloc[i][0].strip().split(' ')
        list1.append(list)
    datanp = np.array(list1)
    datanp = datanp.reshape(-1).astype(int)
    return datanp
```

```
def Predict(y,A):
    z = np.matmul(A, y)       # z = Ay
    return z

if name=='main':
    N = 12
    T = 7
    R = np.array([39, 36, 38, 40, 37, 33, 40])
    TU = np.array([4500, 2500, 2750, 2100, 2500, 2750, 1500])
    TC = np.array([0, 3, 0, 0, 0, 5, 0, 0, 5, 0, 0, 0])
    CP = np.array([240, 120, 40, 60, 50, 160, 80, 100, 180, 60, 40, 70])
    CS = np.array([5, 2, 5, 3, 6, 1.5, 4, 5, 1.7, 3, 2, 3])
    RP = np.zeros((N, N))
    RP[0][1], RP[0][5], RP[0][8], RP[1][2], RP[1][3], RP[1][4], RP[5][6], RP[5][7], RP[8][9],
RP[8][10], RP[8][
        11] = 3, 4, 5, 6, 8, 2, 2, 4, 8, 2, 12


    A = getA('question4-1.csv')
    SOI = getSOI('question4-2.csv')
    Y = np.array([37, 41, 39, 41, 36, 32, 44])
    R = Predict(Y,A).astype(int)


    wpcrp = WPCRP("WPCRP 4-3")
    paramSet = T, N, R, TU, TC, CP, CS, RP, SOI
    wpcrp.solve(paramSet)
    print(wpcrp.GetSol())
    wpcrp.write("question4-3.csv")
```